QUERY CLAUSES

```
-- • Returned by your select statement
-- • Created by your insert statement
-- • Modified by your update statement
-- • Removed by your delete statement
```

-- **select** Determines which columns to include in the query’s result set

-- **from** Identifies the tables from which to retrieve data and how the tables should be joined

-- **where** Filters out unwanted data

-- **order by** Sorts the rows of the final result set by one or more columns.

```
-- The select Clause.

USE sakila;

-- Show me all the columns and all the rows in the language table.

SELECT
    *
FROM
    language;



-- You can explicitly name the columns you are interested in, such as:

SELECT
    name, language_id
FROM
    language;

SELECT
    name
FROM
    language;

/*The select clause determines which of all possible columns should be included in the query’s result set.*/

/*The next query demonstrates the use of a table column, a literal, an expression, and a built-in function call in a single query against the language table, We cover expressions and built-in functions in detail later this is just a demo*/
```

```
SELECT

    language_id,

    'common' language_usage,

    language_id * 3.1415927 lang_pi_value,

    UPPER(name) language_name

FROM

    language;


/* When you call a built in function and doesn’t retrieve data from
any tables, there is no need for a from clause.*/

SELECT VERSION(), USER(), DATABASE();
```

Column Aliases :

```
SELECT
    UPPER(name) language_name
FROM
    language;

SELECT
    UPPER(name) language_name, language_id * 3.14 lang_pi_value
FROM
    language;

--Here language_name and lang_pi_value are aliases.

/*In order to make your column aliases stand out even more, you also
have the option of using the as keyword before the alias name, as
in:*/

SELECT
    UPPER(name) AS language_name
FROM
    language;
```

Removing Duplicates

```
SELECT
    actor_id
FROM
    film_actor
ORDER BY actor_id;

-- Since some actors appeared in more than one film, you will see the same actor ID multiple times.

SELECT DISTINCT
    actor_id
FROM
    film_actor
ORDER BY actor_id;

/*The result set now contains 200 rows, one for each distinct actor, rather than 5,462 rows, one for each film appearance by an actor.*/

desc actor;

SELECT
    actor_id
FROM
    actor
ORDER BY actor_id;

/*Keep in mind that generating a distinct set of results requires the data to be sorted, which can be time consuming for large result sets. Don’t fall into the trap of using distinct just to be sure there are no duplicates; instead, take the time to understand the data you are working with so that you will know whether duplicates are possible.*/
```

The from Clause

/*The from clause defines the tables used by a query, along with the means of linking the tables together.*/

/*• Permanent tables (i.e., created using the create table statement)

• Derived tables (i.e., rows returned by a subquery and held in memory)

• Temporary tables (i.e., volatile data held in memory)

• Virtual tables (i.e., created using the create view statement)*/

/* **Derived Table -**

A subquery is a query contained within another query. Subqueries are surrounded by

parentheses and can be found in various parts of a select statement; within the from

clause*

```
SELECT
    CONCAT(cust.last_name, ',', cust.first_name) AS full_name
FROM
    (SELECT
        first_name, last_name
    FROM
        customer
    WHERE
        first_name = 'JESSIE') cust;
```

Temporary Table -

Every relational database allows the ability to define volatile, or temporary, tables. These tables

look just like permanent tables, but any data inserted into a temporary table will disappear at some

point (generally at the end of a transaction or when your database session is closed). he exception is

Oracle Database, which keeps the definition of the temporary table available for future sessions.

Views -

A view is a query that is stored in the data dictionary. It looks and acts like a table, but there is no data associated with a view (this is why it is called a virtual table).

The Where Clause

The where clause is the mechanism for filtering out unwanted rows from your result set.

/*For example, perhaps you are interested in renting a film but you are only interested in movies rated G that can be kept for at least a week. The following query employs a where clause to retrieve only the films meeting these criteria:*/

```
SELECT

    title

FROM

    film

WHERE

    rating = 'G' and language_id = 1;


SELECT

    title

FROM

    film

WHERE

    rating = 'G' and rental_duration > 5;


SELECT

    title

FROM

    film

WHERE

    rating = 'G' or rental_duration > 7;
```

```sql
/*This where clause contains two filter conditions, but you can
include as many conditions as are required; individual conditions are
separated using operators such as and, or, and not*/

-- Using both and and or operators in your where clause?  You should
use parentheses to group conditions together.


SELECT

    title, rating, rental_duration

FROM

    film

WHERE

    (rating = 'G' AND rental_duration >= 7)

        OR (rating = 'PG-13' AND rental_duration < 4);
```

The order by Clause

The order by clause is the mechanism for sorting your result set using either raw column data or expressions based on column data.

```
desc film;


SELECT
    title, release_year
FROM
    film
WHERE
    title = 'ALABAMA DEVIL'
ORDER BY title;


/*When sorting, you have the option of specifying ascending or descending order via the asc and desc keywords. The default is ascending*/


desc actor;


SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    first_name = 'Nick'
ORDER BY actor_id;
```

```sql
SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    first_name = 'Nick'
ORDER BY actor_id DESC;
```

-- Sorting via Numeric Placeholders

```sql
SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    first_name = 'Nick'
ORDER BY 1 DESC;

SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    first_name = 'Nick'
ORDER BY 3 DESC;
```

```
SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    first_name = 'Nick'
ORDER BY 3;
```

Assignments -

Exercise - 1 Retrieve the actor ID, first name, and last name for all actors. Sort by last name and then by first name.

Exercise - 2 Retrieve the actor ID, first name, and last name for all actors whose last name equals 'WILLIAMS' or 'DAVIS'.

Exercise 3 - Write a query against the rental table that returns the IDs of the customers who rented a film on July 5, 2005 (use the rental.rental_date column, and you can use the date() function to ignore the time component). Include a single row for each distinct customer ID.

Exercise 4 - Fill in the blanks (denoted by <numbers>) for this multitable query to achieve the following

results:

```
mysql> SELECT c.email, r.return_date
-> FROM customer c
-> INNER JOIN rental <1>
-> ON c.customer_id = <2>
-> WHERE date(r.rental_date) = '2005-06-14'
-> ORDER BY <3> <4>;
```

```
+----------------------------------------+---------------------+
| email                                  | return_date         |
+----------------------------------------+---------------------+
| DANIEL.CABRAL@sakilacustomer.org       | 2005-06-23 22:00:38 |
| TERRANCE.ROUSH@sakilacustomer.org      | 2005-06-23 21:53:46 |
| MIRIAM.MCKINNEY@sakilacustomer.org     | 2005-06-21 17:12:08 |
| GWENDOLYN.MAY@sakilacustomer.org       | 2005-06-20 02:40:27 |
| JEANETTE.GREENE@sakilacustomer.org     | 2005-06-19 23:26:46 |
| HERMAN.DEVORE@sakilacustomer.org       | 2005-06-19 03:20:09 |
| JEFFERY.PINSON@sakilacustomer.org      | 2005-06-18 21:37:33 |
| MATTHEW.MAHAN@sakilacustomer.org       | 2005-06-18 05:18:58 |
| MINNIE.ROMERO@sakilacustomer.org       | 2005-06-18 01:58:34 |
| SONIA.GREGORY@sakilacustomer.org       | 2005-06-17 21:44:11 |
| TERRENCE.GUNDERSON@sakilacustomer.org  | 2005-06-17 05:28:35 |
| ELMER.NOE@sakilacustomer.org           | 2005-06-17 02:11:13 |
| JOYCE.EDWARDS@sakilacustomer.org       | 2005-06-16 21:00:26 |
| AMBER.DIXON@sakilacustomer.org         | 2005-06-16 04:02:56 |
| CHARLES.KOWALSKI@sakilacustomer.org    | 2005-06-16 02:26:34 |
| CATHERINE.CAMPBELL@sakilacustomer.org  | 2005-06-15 20:43:03 |
+----------------------------------------+---------------------+
```

Assignments Solution –

/* **Exercise - 1** Retrieve the actor ID, first name, and last name for all actors. Sort by last name and then by first name.*/

```
SELECT
    actor_id, first_name, last_name
FROM
    actor
ORDER BY last_name,first_name;

-- or

SELECT
    actor_id, first_name, last_name
FROM
    actor
ORDER BY 3,2;
```

```sql
/* Exercise - 2 Retrieve the actor ID, first name, and last name for
all actors whose last name equals 'WILLIAMS' or 'DAVIS'.*/


SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    last_name = 'WILLIAMS' OR last_name = 'DAVIS';


-- or


SELECT
    actor_id, first_name, last_name
FROM
    actor
WHERE
    last_name IN ('WILLIAMS' , 'DAVIS');
```

```
/* **Exercise 3 -** Write a query against the rental table that returns the IDs of the customers who rented a film on July 5, 2005 (use the rental.rental_date column, and you can use the date() function to ignore the time component). Include a single row for each distinct customer ID.*/
```

```
SELECT DISTINCT

    customer_id

FROM

    rental

WHERE

    DATE(rental_date) = '2005-07-05';
```

/* **Exercise 4 -** Fill in the blanks (denoted by <numbers>) for this multitable query to achieve the following

results:

```
mysql> SELECT c.email, r.return_date

-> FROM customer c

-> INNER JOIN rental <1>

-> ON c.customer_id = <2>

-> WHERE date(r.rental_date) = '2005-06-14'

-> ORDER BY <3> <4>;

+----------------------------------------+---------------------+

| email                                  | return_date         |

+----------------------------------------+---------------------+

| DANIEL.CABRAL@sakilacustomer.org       | 2005-06-23 22:00:38 |

| TERRANCE.ROUSH@sakilacustomer.org      | 2005-06-23 21:53:46 |

| MIRIAM.MCKINNEY@sakilacustomer.org     | 2005-06-21 17:12:08 |

| GWENDOLYN.MAY@sakilacustomer.org       | 2005-06-20 02:40:27 |

| JEANETTE.GREENE@sakilacustomer.org     | 2005-06-19 23:26:46 |

| HERMAN.DEVORE@sakilacustomer.org       | 2005-06-19 03:20:09 |

| JEFFERY.PINSON@sakilacustomer.org      | 2005-06-18 21:37:33 |

| MATTHEW.MAHAN@sakilacustomer.org       | 2005-06-18 05:18:58 |

| MINNIE.ROMERO@sakilacustomer.org       | 2005-06-18 01:58:34 |

| SONIA.GREGORY@sakilacustomer.org       | 2005-06-17 21:44:11 |

| TERRENCE.GUNDERSON@sakilacustomer.org  | 2005-06-17 05:28:35 |

| ELMER.NOE@sakilacustomer.org           | 2005-06-17 02:11:13 |

| JOYCE.EDWARDS@sakilacustomer.org       | 2005-06-16 21:00:26 |

| AMBER.DIXON@sakilacustomer.org         | 2005-06-16 04:02:56 |

| CHARLES.KOWALSKI@sakilacustomer.org    | 2005-06-16 02:26:34 |

| CATHERINE.CAMPBELL@sakilacustomer.org  | 2005-06-15 20:43:03 |

+----------------------------------------+---------------------+
desc rental;
desc customer;


SELECT
    c.email, r.return_date
FROM
    customer c
        INNER JOIN
    rental AS r ON c.customer_id = r.customer_id
WHERE
    DATE(r.rental_date) = '2005-06-14'
ORDER BY 2 DESC;
```